Virtualization and Fault Tolerance in Cloud Computing

Thesis submitted in partial fulfilment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(*Specialization: Computer Science*)

by

Pranesh Das

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Virtualization and Fault Tolerance in Cloud Computing

Dissertation submitted in

Jun 2013

to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Pranesh Das

(Roll 211CS1270)

under the supervision of

Prof. Pabitra Mohan Khilar

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela – 769 008, India

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Jun 3, 2013

Certificate

This is to certify that the work in the thesis entitled **Virtualization and Fault Tolerance in Cloud Computing** by **Pranesh Das**, bearing roll number **211CS1270**, is a record of an original research work carried out by him under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of **Master of Technology** in **Computer Science and Engineering**. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Prof. Pabitra Mohan Khilar
Dept. of Computer Science and Engineering
National Institute of Technology, Rourkela

Acknowledgment

First of all I would like to thank, my Parents, without their moral support and blessings I wouldn't have been writing this "thesis".

I would like to express my deep sense of respect and gratitude towards my supervisor Prof. Pabitra Mohan Khilar, who has been the guiding force behind this work. Without his unconditional support it wouldn't have been possible. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. I want to thank him for giving me the opportunity to work under him. His invaluable advice and assistance helped me to complete this thesis. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

I am very much indebted to Prof. Ashok Kumar Turuk, Head-CSE, for his continuous encouragement and support. He is always ready to help with a smile. I am also thankful to all the professors of the department for their support. I am really thankful to all my friends. My sincere thanks to everyone who has provided me with kind words, a welcome ear, new ideas, useful criticism, or their invaluable time, I am truly indebted.

I must acknowledge the academic resources that I have got from NIT Rourkela. I would like to thank administrative and technical members of the Department who have been kind enough to advise and help in their respective roles.

Last,but not the least, I would like to dedicate this thesis to my family, for their love, patience, and understanding.

Pranesh Das

Abstract

Fault tolerance in cloud computing is a grand challenge problem now a days. The main fault tolerance issues in cloud computing are detection and recovery. To combat with these problems, many fault tolerance techniques have been designed to reduce the faults. In this research work,a Virtualization and Fault Tolerance (VFT) technique is used to reduce the service time and to increase the system availability. A Cloud Manager (CM) module and a Decision Maker (DM) are used in this scheme to manage the virtualization, load balancing and to handle the faults. In the first step virtualization and load balancing are done and in the second step fault tolerance is achieved by redundancy, checkpointing and fault handler.The main load balancing issues in cloud computing is load calculation and load distribution. To solve these issues, many load balancing techniques have been designed to distribute tasks properly. In this work, a Load Balancing Technique for Virtualization and Fault Tolerance in Cloud Computing (LBVFT) is applied to assign the tasks to the virtual nodes.LBVFT is mainly designed to assign tasks to the virtual nodes depending on the success rates (SR) and the previous load history. In our load assigning technique assignment is done by the load balancer (LB) of cloud manager (CM) module in the basis of higher success rate and lower load of the available nodes.A Randomized Searching Algorithm is designed to select a virtual node.Performance of the Randomized Searching Algorithm lies between Binary Search and Linear Search Algorithms. VFT is mainly designed to provide reactive as well as proactive fault tolerance. In this approach a fault handler is included in the virtualization part. Fault handler blocks the unrecoverable faulty nodes along with their virtual nodes for future requests and removes the temporary software faults from the recoverable faulty nodes and makes them available for the future requests.

***Keywords*:** Fault Tolerance, Cloud Virtualization, Hypervisor, Cloud Computing, Load Balancing, LBVFT, VFT, Cloud Manager, Decision Maker, Fault handler,Success Rate, Load, Algorithm, complexity analysis, Binary Search, Linear Search.

Contents

List of Figures

List of Tables

List of Abbreviations

- VFT : Virtualization and Fault Tolerance
- CM: Cloud Manager
- DM: Decision Maker
- LBVFT: Load Balancing Technique for Virtualization and Fault Tolerance
- SR: Success Rate
- LB: Load Balancer

Chapter 1

Introduction

1.1 Introduction

Fault tolerance is an approach where a system continues to success even if there is a fault [2,3]. Although there are number of fault tolerant models or techniques are available but still fault tolerance in cloud computing is a challenging task [4,5,6,7,8,9,11]. Because of the very large infrastructure of cloud and the increasing demand of services an effective fault tolerant technique for cloud computing is required. In our proposed model fault tolerance is integrated with the cloud virtualization [22,23]. The basic mechanism to achieve the fault tolerance is replication or redundancy. We have performed this replication in form of software variants running on multiple virtual machines. We have presented a virtualization approach with the help of hypervisor where the load balancer takes high responsibility by distributing loads only to those virtual nodes whose corresponding physical servers have a good performance history. To measure the performance history of a physical server we have used success rate. If n1=number of times a physical server gives successful results and n2=total number of times requests sent to that server, then the Success Rate SR =n1/n2, where n1<=n2.
An issue in distributed scheduling is load balancing which tries to distribute the

tasks to be executed among the resources of the system [31,32,33]. Load balancing can be achieved either locally or in a distributed fashion. Distributing tasks across a communication medium is sometimes referred to as the resource allocation problem. Resource allocation actually refers to scheduling multiple resources. Here a load assigning technique LBVFT for the proposed VFT model is proposed. The proposed load assigning scheme for the VFT model assigns a task to the available virtual nodes depending on their success rates and the load history. Because of the very large infrastructure of cloud and the increasing demand of services an effective fault tolerant technique for cloud computing is required and for which an effective load balancing approach is required. In the VFT model the load balancer takes high responsibility by distributing loads only to those virtual nodes whose corresponding physical servers have a good performance history.

A randomized algorithm is one that receives, in addition to its input data, a stream of random bits that it can use for the purpose of making random choices. Even for a fixed input, different runs of a randomized algorithm may give different results [38,39].In our proposed Randomized Searching Algorithm we choose a position randomly in each iteration and then the element of that position is compared with the key item.The performance of the Proposed Algorithm lies between Binary Search and Linear Search.

Chapter Organization: Section 1.1 describes the introduction,section 1.2 gives a brief view on virtualization,section 1.3 focuses on fault tolerance and its different types,section 1.4 gives the load balancing concept,section 1.5 basic concepts of cloud computing,section 1.6 motivation,section 1.7 objectives,section 1.8 thesis organization and section 1.9 conludes the chapter.

1.2 Virtualization

Virtualization is an emerging IT paradigm that separates computing functions and technology implementations from physical hardware. Virtualization technology

allows servers and storage devices to be shared and utilization be increased. Applications can be easily migrated from one physical server to another.

1.3 Fault Tolerance

There are various faults which can occur in cloud computing .Based on fault tolerance policies various fault tolerance techniques can be used that can either be task level or workflow level[9].

1.3.1 Reactive Fault Tolerance

Reactive fault tolerance policies reduce the effect of failures on application execution when the failure effectively occurs. There are various techniques which are based on these policies like Checkpoint/Restart, Replay and Retry and so on.

1. **Check pointing/ Restart:**
   When a task fails, it is allowed to be restarted from the recently checked pointed state rather than from the beginning. It is an efficient task level fault tolerance technique for long running applications [2].

2. **Replication:**
   Various task replicas are run on different resources, for the execution to succeed till the entire replicated task is not crashed. It can be implemented using tools like HAProxy, Hadoop and AmazonEc2 etc.

3. **Job Migration:**
   During failure of any task, it can be migrated to another machine. This technique can be implemented by using HAProxy.

4. **Task Resubmission:**
   It is the most widely used fault tolerance technique in current scientific

workflow systems. Whenever a failed task is detected, it is resubmitted either to the same or to a different resource at runtime.

5. **User defined exception handling:**
   In this user specifies the particular treatment of a task failure for workflows.

6. **Rescue workflow:**
   This technique allows the workflow to continue even if the task fails until it becomes impossible to move forward without catering the failed task.

1.3.2 Proactive Fault Tolerance

The principle of proactive fault tolerance policies is to avoid recovery from faults, errors and failures by predicting them and proactively replace the suspected components other working components. Some of the techniques which are based on these policies are Preemptive migration, Software Rejuvenation etc.

1. **Software Rejuvenation:**
   It is a technique that designs the system for periodic reboots. It restarts the system with clean state.

2. **Proactive Fault Tolerance using Self Healing:**
   When multiple instances of an application are running on multiple virtual machines, it automatically handles failure of application instances.

3. **Proactive Fault Tolerance using Preemptive Migration:**
   Preemptive Migration relies on a feedback-loop control mechanism where application is constantly monitored and analyzed.

1.4 Load Balancing

Load balancing is a computer networking method for distributing workloads across multiple computers or a computer cluster, network links, central processing units,

disk drives, or other resources. Successful load balancing optimizes resource use, maximizes throughput, minimizes response time, and avoids overload. Using multiple components with load balancing instead of a single component may increase reliability through redundancy. Load balancing is usually provided by dedicated software or hardware, such as a multilayer switch or a Domain Name System server Process.

1.5 Basic Concepts of Cloud Computing

Cloud computing refers to applications and services that run on a distributed network using virtualized resources and accessed by common internet protocols and networking standards.It is distinguished by the notion that resources are virtual and limitless and that details of the physical systems on which the software runs are abstracted from the user.
The use of the word "cloud" makes reference to the two essential concepts:

Abstraction:
Cloud computing abstracts the details of system implementation from users and developers.Applications run physical systems that are not specified,data is stored in location that are unknown,adminitration of system are outsourced to others and access by users is ubiquitous.

Virtualization:
Cloud computing virtualizes systems by pooling and sharing resources.Systems and storage can be provisioned as needed from a centralized infrastructure,costs are accessed on a metered basis,multi tenancy is enabled and resources are scalable with agility.

1.5.1 Cloud Types

Most people separate cloud computing into two distinct sets of models:

Deployment models:

This refers to the location and management of the cloud's infrastructure.Example-Public cloud,Private cloud,Hybrid cloud and community cloud.

Service models:

This consists of the particular types of services that we can access on a cloud computing platform.

- Infrastructure as a service.
- Platform as a service.
- Software as a service.

1.5.2 Advantages

Lower computer costs:

Not necessary to have high-powered computers to access web applications. Even with cheaper computer also can give efficient results because data is stored in the web not with us.

Improved performance:

Everything is run in cloud so our computer doesnt have to take much effort to run applications. As a result, performance will be improved automatically.

Unlimited storage capacity:

Storage is also one kind of service provided by the Cloud, so there is no limit to store data (based on the service provider).

Device independence:

The actual documents are in the Cloud, so we can access it wherever we are.

1.5.3 Disadvantages

Requires a constant High speed Internet connection:

To get benefit from this we need to have always a high speed Internet connection.

Stored data might not be secure:

There is no guarantee that your data stored is in cloud is securely protected. Intruders may access to your vital data at any time.

1.6 Motivation

After the study, we found that the main fault tolerance issues in cloud computing are detection and recovery. To combat with these problems, many fault tolerance techniques have been designed to reduce the faults.But due to its virtualization and internet based service providing behaviour fault tolerance in cloud computing is still a big challenge. Our proposed model is not only to tolerate faults but also to reduce the chance of future faults by not assigning tasks to virtual nodes of physical servers whose success rates are very low.

1.7 Objective

Our objectives are:

- to design a fault tolerance model which can provide reactive and proactive fault tolerance in cloud computing.
- to design a load balancing technique for the proposed model to assign tasks to the virtual nodes.
- to design a searching algorithm to search a virtual node for task assignment.

1.8 Thesis Organization

The rest of the thesis is organized as follows: In Chapter 2,we give the literature review, In Chapter 3, we proposed A Virtualization and Fault Tolerance Model for Cloud Computing(VFT). In Chapter 4, we proposed A Load Balancing Technique for Virtualization and Fault Tolerance in Cloud Computing(LBVFT) for the VFT model. In Chapter 5, we proposed A Randomized Searching Algorithm to select a node for task assignment. Finally in Chapter 6, we conclude our thesis.

1.9 Conclusion

In this chapter the background concepts for the remaining chapters are given.The basic concepts of cloud computing,fault tolerance,load balancing are briefly discussed here.Then the objectives and motivation of the thesis work are discussed.

Chapter 2

Literature Survey

2.1 Introduction

In this chapter we briefly discuss the research conducted so far in fault tolerance and load balancing. There are number of fault tolerance and load balancing algorithms exist, but they are mostly based on either reactive or proactive fault tolerance but not both.

2.2 Literature Survey

A lot of work has been done in the area of fault tolerance and load balancing for cloud computing. But due to its virtualization and internet based service providing behaviour fault tolerance and load balancing in cloud computing are still a big challenge. Many researchers have given various fault tolerance techniques and strategies in [4,5,6,7,8,9,10,11,12,20,21,24,25]. Dilbag Singh and Jaswinder Singh in [4] have given failover strategies for cloud computing using integrated checkpointing algorithms. Sheheryar Malik and Fabrice Huet in [10] have given an approach for adaptive fault tolerance in real time cloud computing.Many researchers have given various load balancing techniques in [30,31,32,33,34,35,37].The proposed LBVFT technique distributes loads in a smart way by considering the success rates and the

loads history of the available virtual nodes. Thus the LBVFT helps the VFT model to tolerate not only faults but also reduce the chance of future faults by not assigning tasks to virtual nodes of physical servers whose success rates are very low and loads are very high.

Chapter Organization Section 1.1 gives the introduction,section 1.2 related work or literature survey is discussed and section 1.3 concludes the chapter.

2.3 Conclusion

After the analysis and survey on the related works and some selected research works on fault tolerance in cloud computing,load balancing and virtualization it is seen that fault tolerance in cloud computing is a challenging task.This thesis work proposes a model for fault tolerance.

Chapter 3

Proposed Virtualization and Fault Tolerance Model(VFT) for Cloud Computing

3.1 Introduction

The Virtualization and Fault Tolerance (VFT) model is proposed here which provides the reactive fault tolerance on cloud infrastructure. This scheme tolerates the faults on the basis of Success Rate (SR) ($0 < SR <= 1$) of each virtual node's physical server. A virtual node is selected for computation on the basis of SR of its corresponding physical server and can be removed, if the selected nodes physical server does not perform well. Our model consists of two main modules Cloud Manager (CM) module and Decision Maker (DM) module .The model is shown in figure 1.

Chapter Organization: In this chapter section 1.1 introduces the chapter,section 1.2 working of the proposed model,section 1.3 fault tolerance technique,and section 1.4 success rate analysis of the nodes are given.Section 1.5 simulation result,section 1.6 evaluation of the simulated results and section 1.7

concludes the chapter.

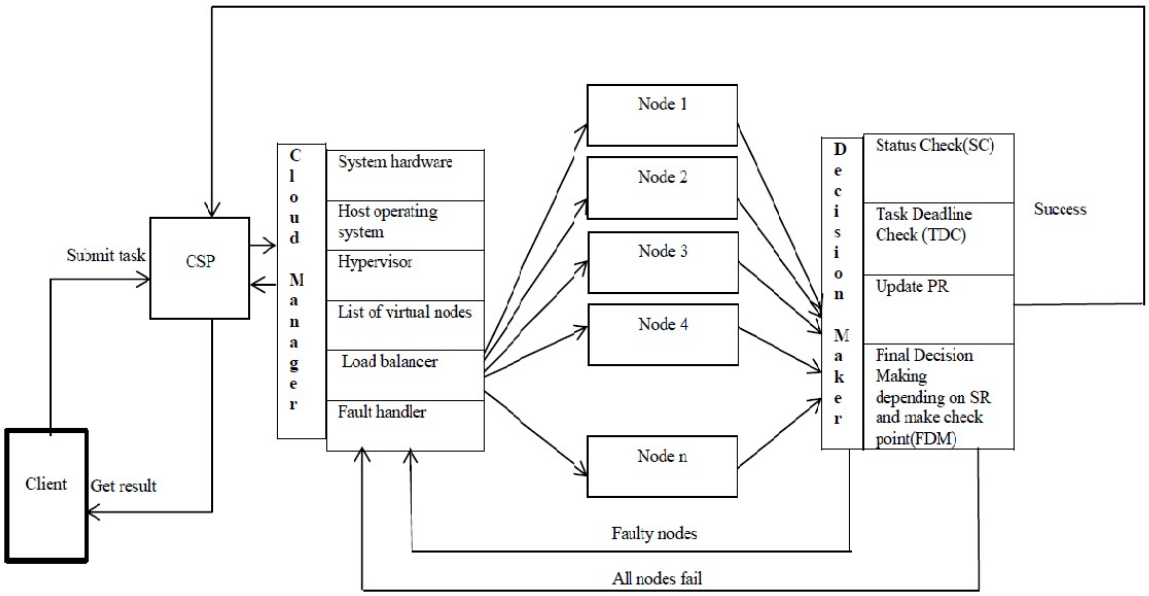

Figure 3.1: Proposed VFT Model

3.2 Working of the Proposed VFT Model

In our proposed scenario a client submits a task to the Cloud Service Provider (CSP).The CSP then submits it to the Cloud Manager (CM).

3.2.1 Cloud Manager(CM)

Cloud Manager is included in the cloud architecture. It performs the virtualization with the help of Hypervisor. Hypervisor is a low level program which creates a virtual environment and provides system resource access to virtual machines. When virtual nodes are created from the available resources of the physical servers (System Hardware) then Hypervisor maintains a record of which virtual node belongs to which physical node. Resources of a single physical server can be used to create set

of virtual nodes. A Performance Record (PR) table is maintained containing the servers ids, virtual nodes ids and Success Rate (SR) to identify the virtual nodes and to keep record of the number of times tasks are assigned to the virtual nodes of a particular server and gets successful result from those virtual nodes of the corresponding physical server. Here the SR is the success rate of a physical node whose resources are used for virtualization. The PR table is also available to the Load Balancer.

3.2.2 Load Balancer

Load Balancer distributes the loads based on the Performance Record (PR) of the physical systems that are used for virtualization .Load Balancer assigns tasks to the virtual nodes with the help of load balancing algorithms[26], [27], [28]. The Load Balancer always assigns the tasks to those virtual nodes whose corresponding physical servers are having good SR.

3.2.3 Fault Handler

Fault Handler takes the responsibility when a virtual node is found faulty due to some recoverable temporary software faults in the CM or due to some transient faults occurred in the remote physical server of the corresponding virtual node .The PR table gets updated .If there is no virtual node executing under that physical server then Fault Handler immediately restart the remote server and informs the Load balancer not to assign any task to the virtual nodes of the corresponding server .The Fault Handler may apply some transient fault detection and recovery technique[4],[5].If the fault handling is done successfully then the virtual node or virtual nodes of that physical server are made available for future request.Results that are executed by the virtual nodes are submitted to the Decision Maker (DM) module.

3.2.4 Decision Maker(DM)

Decision Maker is also included in the cloud architecture like CM. Status Checker (SC) in DM checks the status of each virtual node. If status is success then Task Deadline is checked by TDC sub module in DM. If both Status Checking and Task Deadline checking (TDC) is success then SR of the corresponding node is increased and it goes for Final Decision Making (FDM) sub module. If either SC or TDC is fail then the corresponding virtual machine is not sent to the FDM sub module. If the SC is fail then the node is sent to the Fault Handler for fault detection and recovery. If SC is success but the task is not completed within the time limit then also the node is not considered for final decision making and SR in the PR table is decreased for that node. FDM sub module contains all the virtual nodes that successfully executed the SC and TDC modules. FDM then selects a node with the highest SR value and makes a checkpoint [13], [14], [15], [16], [29]. If more than one node is having same SR value then selects any one randomly. If all nodes are failed then backward recovery is performed with the help of the last successful checkpoint.

3.3 Fault Tolerance Technique

In our proposed VFT model we apply SR computation algorithm for each node (virtual machine) one by one. Initially SR of a node is set to 0.5. This algorithm takes input of two factors, PR table (SR, n1, n2, vmid; serverid)and maxSuccessRate.maxSuccesRate is the maximum SR level. Node SR cannot be more than this level. In our proposed scheme maxSuccessRate is 1. It is really important in a situation , where an initially produces correct results in consecutive cycles, but then fails again and again. In our proposed scheme the SR value cannot be zero because Load Balancer will not assign any task to a virtual node whose corresponding server nodes SR is less than equal to zero.

Success Rate Computation Algorithm:

1. Initially success rate =0.5, n1=1, n2=2
2. n1 is the number of times the virtual node of a particular physical server gives successful results
3. n2 is the number of times the Load Balancer of the cloud manager(CM) assigns tasks to a particular servers virtual node
4. Input maxSuccessRate=1
5. Status of a node is Success if SC and TDC module for that node is success
6. Status of a node is Fail if SC or TDC or both module for that node is fail
7. if (nodeStatus = =Success) /*SC and TDC success*/
   {
   n1=n1+1
   n2=n2+1
   SuccessRate = n1/n2
   Update PR table
   }
8. else
   {

   if(nodeStatus = =Fail) /* SC or TDC or both fail */
   {
   n2=n2+1
   SuccessRate= n1/n2
   Update PR table
   }
   }

9. if (SuccessRate>=maxSuccessRate)
   {
   SuccessRate = maxSuccessRate
   }
10. if(SuccessRate<=0)
    {
    Reject the node and inform the Cloud Manager to add a new node
    }

Decision Technique Algorithm:

1. Initially SuccessRate=0.5
2. Input from TDC:nodeSuccessRate, n=number of nodes with SC and TDC success
3. Input maxSuccessRate
4. if(n= =0)
   {
   Status = fail
   Perform backward recovery with the last successful checkpoint
   }
5. else
   {
   Status =Success
   bestSuccessRate = find SuccessRate of node with highest SuccessRate
   Select the node with bestSuccessRate and send the result to CSP
   Make checkpoint
   }

3.4 Success Rate Analysis

Here we have given a metric analysis to analyse the success rate for different scenarios of four virtual nodes. We have taken 100 computing cycles. In the be- ginning we have assumed that success rate is 0.5. In figure 3.2, figure 3.3, figure 3.4 and figure 3.5, a comparison analysis is given between success and failure scenarios. This comparison is done for 100 computing cycles. In these cycles vm1 continuously succeeded, vm2 continuously failed, vm3 succeeded for first 50 cycles and then failed for remaining 50 cycles and vm4 failed for first 50 cycles and then succeeded for remaining 50 cycles. The increase in success rate after 100 cycles for vm1 is 0.8545, whereas decrease in success rate for vm2 is 0.0545, and increase in success rate for vm3 and vm4 is 0.4.We have assumed that each virtual node belongs to a different physical server. Here we can see that increase in success rate is more than decrease. Hence we can achieve a good performance of the algorithm.

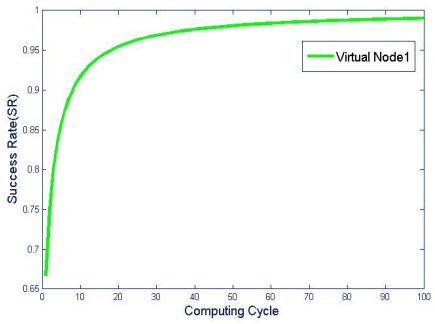

Figure 3.2: Continuous increase in success rate (SR)

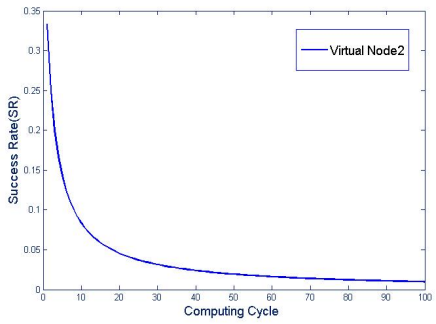

Figure 3.3: Continuous decrease in success rate (SR)

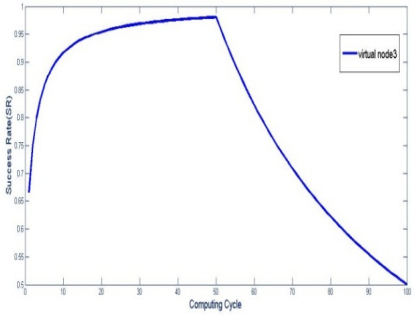

Figure 3.4: Pass to fail shifting

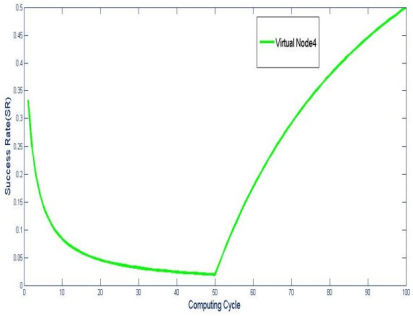

Figure 3.5: Fail to pass shifting

3.5 Simulation Result

We have done the simulation in Cloudsim 2.0 [1], [18], [19] with NetBeans IDE 6.7.1. In this simulation we have created three virtual nodes. Every virtual node executes the same task at a time. The algorithm has 10 computing cycles. The algorithms consist of a series of tasks. Each of these tasks is performed in one computing cycle. Each virtual node may run diverse algorithm.

Simulation Result:

Cycle	Task Deadline	Virtual Node 1				Virtual Node 2				Virtual Node 3				Selected Node
		SC	TDC	Finish Time	SR	SC	TDC	Finish Time	SR	SC	TDC	Finish Time	SR	
Start	-	-		-	0.5	-	-	-	0.5	-		-	0.5	-
1	1700	1	1	1600.0	0.667	1	1	1601.6	0.667	1	1	1610	0.667	1
2	1602	1	1	1600.4	0.75	1	1	1602.0	0.75	1	0	1610.4	0.5	2
3	1601	1	1	1600.8	0.8	1	0	1602.4	0.6	1	0	1610.8	0.4	1
4	1605	1	1	1601.2	0.833	1	1	1602.8	0.667	1	0	1611.2	0.333	1
5	1600	1	0	1602.4	0.714	1	0	1603.2	0.571	1	0	1611.6	0.286	-
6	1900	1	1	1602.0	0.75	1	1	1603.6	0.625	1	1	1612.0	0.375	1
7	1700	1	1	1602.4	0.778	1	1	1604.0	0.667	1	1	1612.4	0.444	1
8	2100	1	1	1603.6	0.8	1	1	1604.4	0.7	1	1	1612.8	0.5	1
9	1700	1	1	1603.2	0.818	0	0	-	0.636	1	1	1613.2	0.545	1
10	2000	1	1	1603.6	0.833	1	1	1605.2	0.666	1	1	1613.6	0.583	1

Here on the above table "1" denotes success and "0" denotes fail in SC and TDC column

Figure 3.6:

Then we have DM module which is receiving the results from the virtual machines

to be processed. This DM is integrated with the CM at the Cloud Service Provider (CSP) [1] site. If a node fails then the system will continued to be executed using the remaining nodes. The system continues its execution until all nodes are failed. After a successful computing cycle a node is selected and a checkpoint is made by the FDM to keep the status of the system for future recovery. Here we have assumed that the following information of the selected virtual nodes is available to us: PR table i.e. value of n1, n2, SR, virtual node id and corresponding server id. Initially n1=1 and n2=2 i.e. SR=0.5 is considered for every node. Tasks deadlines are also given as input. The simulated results are shown on Figure 3.6 and from those results success rate analysis of node1, node2 and node3 is shown on Figure 3.7, Figure 3.8 and Figure 3.9 respectively.

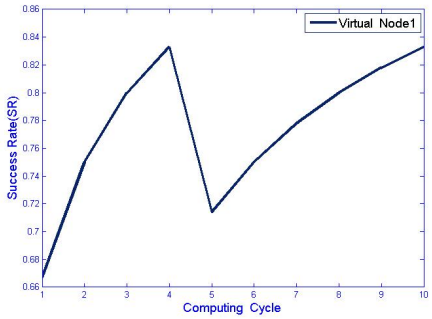

Figure 3.7: Virtual node 1

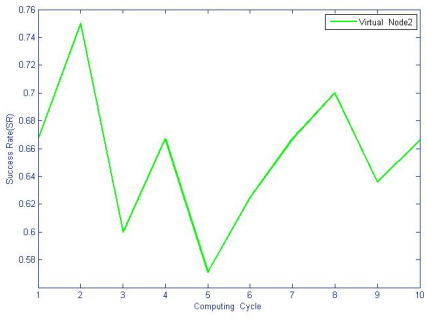

Figure 3.8: Virtual node 2

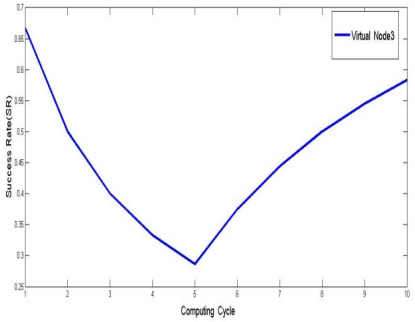

Figure 3.9: Virtual node 3

3.6 Evaluation

In the first cycle, all nodes have the same success rate, so any one can be selected. We have selected node 1.In cycle 2 node 2 is selected. In most of the remaining cycles node 1 is selected as its SR is increases continuously. In cycle 9 virtual node2 did not pass SC and so TDC also failed and node 1 is selected as it has the highest SR. If SC is fail then TDC must be fail , e.g. cycle 9. But if SC passes then TDC may pass or may not. In case of either SC and TDC failure or only SC failure an error message is sent to the fault handler of the CM module and to the TDC sub module. The error handler then tries to repair the fault. TDC received it before time limit. But as there is no result produced, so the status of TDC is also fail. In Figure 3.7, Figure 3.8 and Figure 3.9 success rate(SR)fluctuation is shown.

3.7 Conclusion and Discussion

This paper proposes a smart failover strategy for cloud computing using success rate of the computing nodes and virtualization which include the support of load balancing algorithms and fault handler. A quantitative analysis is given in section V. Performance comparison of existing methods has been made with the purposed method. It has been concluded with the help of performance metrics comparison and success rate analysis from simulated results that the proposed fault tolerant strategy gives a very good performance. In our future work we will work on the Fault Handler and Load Balancer sub modules of CM module in order to make the model more fault tolerant.

Chapter 4

Proposed Load Balancing Technique for the VFT Model in Cloud Computing(LBVFT)

4.1 Introduction

Here in the proposed load balancing approach when cloud manager (CM) of VFT model gets the request from the cloud service provider (CSP) it creates set of virtual nodes with the help of system hardware, host operating system and hypervisor module. System hardware means set of physical servers which are connected by distributed network .CM then gives the responsibility to the load balancer to assign the tasks. As in the VFT model to achieve fault tolerance redundant virtual nodes are used so, the LB will assign the same task to a number of available virtual nodes which are having good SR value and lower load in the performance record table. Thus the same task will be executed in all the selected nodes and the result will be sent to the Decision Maker (DM) module of the VFT model (figure 1).

Chapter organization: Section 1.1 introduces the chapter,section 1.2 provides the working of the proposed load balancing technique,section 1.3 different scenarios of the algorithm,section 1.4 gives the simulation result and section 1.5 concludes the chapter.

4.2 Working of the LBVFT Algorithm

1. Initially success rate (SR)=0.5,maximum SR=1, $0 < SR <= 1$.
2. Input high SR, higher SR, low SR, lower SR, low load, lower load, high load and higher load values.
3. SR=n1/n2
4. n1 is the number of times the virtual node of a particular physical server gives successful results.
5. n2 is the number of times the Load Balancer of the cloud manager(CM) assigns tasks to a particular servers virtual node.
6. Loads of the virtual nodes are given. Loads are generally calculated by the load balancer and is updated and kept in the performance record table of the virtual nodes time to time, which is beyond the scope of this paper.
7. Search for all the available virtual nodes having lower load history and higher SR values in the performance record table.
8. if($(SR == higher || SR == high) \&\& (load == lower)$)
   {
   Select the node
   }
9. else
   {

if($(SR == higher || SR == high) \&\& (load == low)$)

{

Select the node

}

}

10. if($(SR == Higher || SR == High) \&\& (load == higher)$)

    {

    Dont select the node if enough nodes are

    available

    }

11. if($(SR == Higher || SR == High) \&\& (load == high)$)

    {

    Dont select the node if enough nodes are

    available

    }

12. if $((SR == low || SR == lower) \&\& (load == low))$

    {

    Select the node if numbers of nodes are less

    }

13. if $((SR == low || SR == lower) \&\& (load == lower))$

    {

    Select the node if numbers of nodes are less

    }

14. if($(SR == low || SR == lower) \&\& (load == high)$)

    {

Dont select the node

}

15. if($(SR == low || SR == lower) \&\& (load == higher)$)

{

Dont select the node

}

16. Select m nodes from the list of available virtual nodes (m is the user input; m-1 is the number of redundant nodes).

17. Submit the task to all the selected m nodes.

18. The tasks are then executed by the allocated virtual nodes and the results are sent to the decision maker (DM) module of the VFT model.

4.3 Different Scenarios

1. **High or higher success rate and low or lower load:** In this case a node is better one for selection and the node is selected. As the load is low and the SR value is high, there is a less chance of future fault.

2. **High or higher success rate and high or higher load:** If a node having this property is selected then there is a chance of future fault as the load is high. So, this kind of nodes is avoided for selection if sufficient numbers of virtual nodes are available.

3. **Low or lower success rate and high or higher load** A node having this feature is not selected, because the SR value is low as well as load is high.

4. **Low or lower success rate and low or lower load** In this case a node may be selected if sufficient numbers of nodes are not available.

4.4 Simulation Results and Discussion

Simulation is done in Cloudsim2.0 [9,10] with NetBeans IDE 6.7.1.The simulation result that is obtained from VFT model [1] is given in Table 4.4.1.In the resultant table L1, L2.L33 are the loads of the virtual nodes at that instant of time.

Table 4.4.1: Simulation Result Obtained from the VFT model [1]

Cycle	Task Dead line	Virtual Node 1					Virtual Node 2					Virtual Node 3					Selected Node
		SC	TDC	Finish Time	SR	Load	SC	TDC	Finish Time	SR	Load	SC	TDC	Finish Time	SR	Load	
Start	-	-		-	0.5	L1	-	-	-	0.5	L2	-		-	0.5	L3	-
1	1700	1	1	1600.0	0.667	L4	1	1	1601.6	0.667	L5	1	1	1610	0.667	L6	1
2	1602	1	1	1600.4	0.75	L7	1	1	1602.0	0.75	L8	1	0	1610.4	0.5	L9	2
3	1601	1	1	1600.8	0.8	L10	1	0	1602.4	0.6	L11	1	0	1610.8	0.4	L12	1
4	1605	1	1	1601.2	0.833	L13	1	1	1602.8	0.667	L14	1	0	1611.2	0.333	L15	1
5	1600	1	0	1602.4	0.714	L16	1	0	1603.2	0.571	L17	1	0	1611.6	0.286	L18	-
6	1900	1	1	1602.0	0.75	L19	1	1	1603.6	0.625	L20	1	1	1612.0	0.375	L21	1
7	1700	1	1	1602.4	0.778	L22	1	1	1604.0	0.667	L23	1	1	1612.4	0.444	L24	1
8	2100	1	1	1603.6	0.8	L25	1	1	1604.4	0.7	L26	1	1	1612.8	0.5	L27	1
9	1700	1	1	1603.2	0.818	L28	0	0	-	0.636	L29	1	1	1613.2	0.545	L30	1
10	2000	1	1	1603.6	0.833	L31	1	1	1605.2	0.666	L32	1	1	1613.6	0.583	L33	1

Here on the above table "1" denotes success and "0" denotes fail in SC and TDC column.

Figure 4.1:

From Table 4.4.1 another table Table 4.4.2 is constructed. It is assumed that there are 3 virtual nodes available and their SR values and loads are taken .Here it is considered that there are 10 tasks and for every task 3 virtual nodes are there.Proposed load assigning technique will assign tasks to those nodes which have good SR and load values. It is assumed that number of redundant node is 1.Hence for every task 2 virtual nodes will be selected by the proposed technique which is shown in Table 4.4.2.

Table 4.4.2: Constructed from the above simulated results

Tasks	Vm1		Vm2		Vm3		Selected Nodes
	SR	Load	SR	Load	SR	Load	
Start	0.5	L1	0.5	L2	0.5	L3	-
t1	0.667	L4	0.667	L5	0.667	L6	1,2
t2	0.75	L7	0.75	L8	0.5	L9	1,2
t3	0.8	L10	0.6	L11	0.4	L12	1,2
t4	0.833	L13	0.667	L14	0.333	L15	1,3
t5	0.714	L16	0.571	L17	0.286	L18	1,2
t6	0.75	L19	0.625	L20	0.375	L21	1,2
t7	0.778	L22	0.667	L23	0.444	L24	1,2
t8	0.8	L25	0.7	L26	0.5	L27	2,3
t9	0.818	L28	0.636	L29	0.545	L30	1,2
t10	0.833	L31	0.666	L32	0.583	L33	1,2

Here on the above table L1, L2, L3……L33 are the loads of the virtual nodes at that instant.

Figure 4.2:

4.5 Conclusion

This paper gives a smart load distribution strategy for our virtualization and fault tolerance in cloud computing (VFT) model using success rate of the computing nodes and previous load history. In Table 4.4.2 it is seen that maximum time node 1 and 2 is selected. For task t8 and task t4 although node 1 and node 2 has high SR values still are not selected because it is assumed that loads for those nodes are very high. This task assigning technique helps the VFT model to give a good performance. As only higher SR values and lower loads are considered during virtual node selection hence, there is a very less chance of system failure. Our future work is to provide an efficient load balancing, load migration, load calculation and fault handling technique to make the VFT model more effective.

Chapter 5

Proposed Randomized Searching Algorithm to Select a Node in the VFT Model

5.1 Introduction

In our proposed Randomized Searching Algorithm the main idea is to select positions randomly from the sorted input array and compare elements of those positions with the search key until the match is found or the array is finished. If the key element is found in the randomly selected position p then we stop. If the search key is less than or greater than the element of the chosen position then the searching is continued in the left side or in the right sight of the position respectively. We have done the simulation in MATLAB 7.12.0(R2011a).We have generated the elements of the input array A randomly. We also have generated the search key element K randomly and done the simulation for a large no of times with arrays of different sizes.For a given load or SR value a particular node can be searched in the VFT model using this algorithm only if the performance record table is sorted according to load history or success rate.

Chapter Organization: In chapter section 1.1 gives the introduction,section 1.2 working of the proposed algorithm,section 1.3 gives the results and performance analysis of the algorithm and section 1.4 concludes the chapter.

5.2 Working of the Proposed Algorithm

Begin:

1. lb=lower bound of the sorted input array
2. ub=upper bound of the input array
3. while(lb<=ub)
   {
   p=choose a number randomly between lb and ub
   if(K<A(p))
   {
   ub=p-1
   }
   else
   if(K>A(p))
   {
   lb=p+1
   }
   else
   {
   return p
   }
   }
4. return −1 /* couldn't not find the key */

End

5.3 Results

5.3.1 Performance Analysis of the Proposed Algorithm

Complexity Analysis:

We see that in the first two lines of the working of the proposed RSA before the while loop, 2 primitive operations always get executed (two assignments). However, since these operations happen no matter what the input is, we will ignore them for now. Focusing our attention on the while loop, we see that each time the program enters the while loop, we execute 3+k primitive operations (a <= comparison, k number of operations to chose a random number between lb and ub, an array index, and a < comparison), before the program might branch depending on the result of the first if statement.

Depending on the result of the conditional, the program will execute different numbers of primitive operations. If key > A[p], then the program executes 2 more primitive operations. If key < A[p], then the program executes 4 more primitive operations (an array index and a < comparison in the next if, and then a subtraction and assignment). If A[p] = = key, then we execute 3 more primitive operations (2 operations for the if and then 1 operation to return position p).As k is fix for each iteration so we neglect k now. In other words, the number of primitive operations executed in an iteration of the loop is

$$\begin{aligned} &5, \text{ if key} < \text{A[p]} \\ &6 \text{ , if A[p]} = = \text{key, and} \\ &7 \text{ , if key} > \text{A[P]}. \end{aligned}$$

We can now construct a tree that summarizes how many operations are executed in the while loop, depending on the number of times the while loop is executed and the result of the comparisons between A[p] and key in each iteration. Each node in

the tree represents the exit from the algorithm because A[p] = = key. A node is a left child of its parent if in the previous iteration of the while loop, the search was restricted to the left part of the subarray (that is, when A[P] > key). A node is a right child of its parent if in the previous iteration of the while loop, the search was restricted to the right part of the sub array (that is, when A[p] < key).
Here are the first three levels of the tree: In case of Binary Search algorithm height

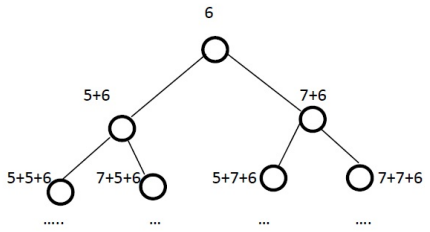

Figure 5.1: No.of operations as a tree structure

of the left sub tree and the height of the right sub tree are equal and we can easily proof that the average case time complexity for binary search is log n.Because the number of nodes are equal in both the left and right sub trees and total number of nodes on each level can be easily computed.But in our proposed algorithm as the position is randomly chosen so the height of left sub tree and the height of right sub tree may not be equal.
Hence to analyse the average case behaviour we have done a simulation on our proposed algorithm. We have done the simulation on the basis of no.of element versus no.of comparisons instead of number operations.In our described algorithm on section 0.4.2 the relevant number of comparisons(one <= comparison,one < and one >) executed in an iteration of the loop is:

2, if key < A[p]

3 , if A[p] = = key, and

3 , if key > A[P].

We have simulated the proposed algorithm in MATLAB 7.12.0(R2011a).In the simulation we have generated the elements of the arrays randomly and also the key.The simulation is done for 50 times with different number of elements ranges from 10 to 500.The simulated result for average case is shown below:

Table 5.1: Simulated Result for Average Case(no.of elements and no.of comparisons)

No.of elements	Compar isons	No.of elements	Compari sons	No.of elements	Compari sons	No.of elements	Compari sons	No.of elements
10	7	110	14	210	24	310	22	410
20	14	120	18	220	17	320	29	420
30	20	130	18	230	30	330	22	430
40	12	140	24	240	41	340	25	440
50	24	150	25	250	18	350	28	450
60	8	160	32	260	21	360	19	460
70	12	170	20	270	22	370	33	470
80	33	180	23	280	15	380	30	480
90	26	190	23	290	18	390	30	490
100	25	200	28	300	19	400	15	500

From the above table Table 1 we see that in average approximately 3 l0g n (base 2) comparisons are required to find the key item for the different set of elements.Hence we can say that approximately the time complexity of our proposed Randomized Searching Algorithm is O(3 log n) in Average Case.

For **Best Case** every time we find the key in the first attempt i.e in the first loop and only 3 comparison are performed for every set of elements.
In **Worst Case** key item is not in the input array i.e in every iteration maximum number of comparisons are performed.The simulation is done in MATLAB 7.12.0(R2011a) for the worst case analysis.Here also we have generated the keys and the elements randomly and done the simulation for 50 times with different number of elements ranges from 10 to 500.The simulated result for **Worst Case** is shown below:

Table 5.2: Simulated Result for Worst Case(no.of elements and no.of comparisons)

No.of elements	Comparisons	No.of elements	Comparisons	No.of elements	Comparisons	No.of elements	Comparisons	No.of elements
10	9	110	27	210	31	310	21	410
20	18	120	20	220	23	320	24	420
30	10	130	14	230	14	330	26	430
40	18	140	20	240	20	340	21	440
50	16	150	10	250	25	350	20	450
60	27	160	22	260	26	360	38	460
70	16	170	19	270	26	370	28	470
80	18	180	32	280	22	380	23	480
90	20	190	13	290	29	390	15	490
100	15	200	14	300	25	400	15	500

From the above table Table 2 we observe that approximately the same number of comparisons are required to find the key items for the different set of elements as like as the average case.Hence we can say that in Worst Case also approximately the time complexity is O(3 log n).

Efficiency Graphs Compared to Binary Search and Linear Search Algorithms:

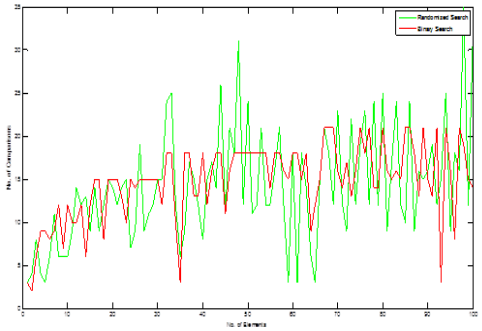

Figure 5.2: Average Case behaviour of the proposed Algorithm and the existing Binary Search Algorithm

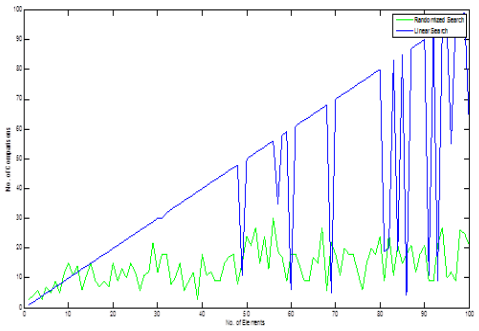

Figure 5.3: Average Case Behaviour of the Proposed Algorithm and the existing Linear Search Algorithm

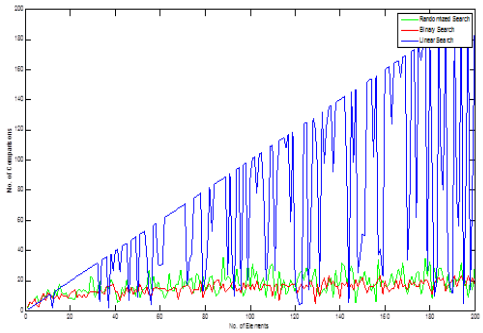

Figure 5.4: Average Case Performance Analysis of the Proposed Algorithm, Binary Search and Linear Search Algorithms.

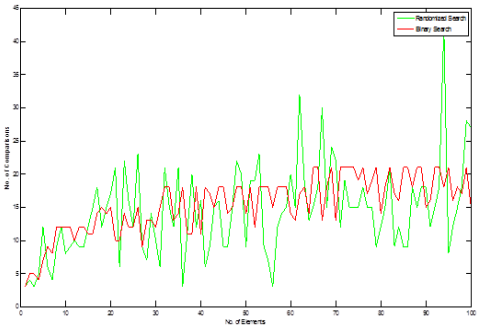

Figure 5.5: Worst Case Behaviour of the Proposed Randomized Search Algorithm and the existing Binary Search Algorithm.

Here on the above figures we have shown the performance analysis of our proposed algorithm graphically. In figure 5.2 we see that our Proposed Algorithm is performing approximately same as Binary Search in average case. During simulation it is seen that some times our proposed algorithm gives better result then the Binary Search. Here we did not get smooth graph because of randomization. From figure 5.3 we see that our proposed algorithm is giving better result then the existing linear Search Algorithm (although linear search is applied on unsorted array).
In figure 5.4 average case behaviour of Linear Search, Binary Search and RSA is shown. In figure 5.5 worst case analysis is done. It is seen that average case and worst behaviour are nearly same for our Proposed RSA Algorithm.

5.4 Conclusion

Performance of the proposed RSA algorithm lies between Binary Search and Linear Search Algorithms. With respect to Linear Search our algorithm gives better performance although Linear Search is applied on unsorted data. Due to randomization we can get an average result every time. We will improve the performance of the algorithm by designing a proper algorithm to choose a number randomly.

Chapter 6

Conclusion and Scope of Future Work

This paper proposes a smart failover strategy for cloud computing using success rate of the computing nodes and virtualization which include the support of load balancing algorithms and fault handler. A quantitative analysis is given in section V. Performance comparison of existing methods has been made with the purposed method. It has been concluded with the help of performance metrics comparison and success rate analysis from simulated results that the proposed fault tolerant strategy gives a very good performance. In our future work we will work on the Fault Handler and Load Balancer sub modules of CM module in order to make the model more fault tolerant. This paper gives a smart load distribution strategy for our virtualization and fault tolerance in cloud computing (VFT) model using success rate of the com- puting nodes and previous load history. In Table 2 it is seen that maximum time node 1 and 2 is selected. For task t8 and task t4 although node 1 and node 2 has high SR values still are not selected because we assumed that loads for those nodes are very high.This task assigning technique helps the VFT model to give a good performance. As only higher SR values and lower loads are considered during virtual node selection hence, there is a very less chance of system failure. Our

future work is to provide an efficient load balancing, load migration, load calculation and fault handling technique to make the VFT model more effective. Performance of the proposed RSA algorithm lies between Binary Search and 41 Linear Search Algorithms. With respect to Linear Search our algorithm gives better performance although Linear Search is applied on unsorted data. Due to randomization we can get an average result every time. We will improve the performance of the algorithm by designing a proper algorithm to choose a number randomly.

Bibliography

[1] http://www.cloudbus.org/cloudsim/

[2] Barrie Sosinsky, "Cloud Computing ", *Wiley India Edition*, ISBN: 978-81-265-2980-3.

[3] Reese, G., "Cloud Application Architectures: Building Applications and Infrastructure in the cloud (Theory in Practice) ", *OReilly Media, 1st Ed.*, 2009 pp 30-46.

[4] Dilbag Singh, Jaswinder Singh, Amit Chhabra, "High Availability of Clouds: Failover Strategies for Cloud Computing using Integrated Checkpointing Algorithms", *IEEE International Conference on Communication Systems and Network Technologies*, 2012.

[5] G. A. Gibson, B. Schroeder, and J. Digney, "Failure Tolerance in Petascale Computers", *CTWatchQuarterly*, vol 3,no 4, November 2007.

[6] Ifeanyi P. Egwutuoha, Shiping Chen, David Levy, Bran Selic, "A Fault Tolerance Framework for High Performance Computing in Cloud", *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2012.

[7] Ghassem Miremadi, Johan Karlsson, Ulf Gunnejlo, Jan Torin, "Two Software Techniques for On-line Error Detection", *Twenty-Second*

International Symposium on Computing Processing (Hardware/Software), 10.1109/FTCS.1992.243622.

[8] Dimitris Gizopoulos, Mihalis Psarakis, Sarita V. Adve, Pradeep Ramachandran, Siva Kumar Sastry Hari, Daniel Sorin, Albert Meixner, Arijit Biswas, and Xavier Vera, "Architectures for Online Error Detection and Recovery in Multicore Processors", *Proceedings of the Design, Automation and Test in Europe (DATE)*, March 2011.

[9] Anju Bala, Inderveer Chana , "Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing", *IJCSI International Journal of Computer Science Issues*, vol. 9, Issue 1, no 1, January 2012.

[10] Sheheryar Malik, Fabrice Huet, Adaptive Fault Tolerance in Real Time Cloud Computing", *IEEE World Congress on Services*, 2011.

[11] Wenbing Zhao, P. M. Melliar-Smith and L. E. Moser, "Fault Tolerance Middleware for Cloud Computing", *IEEE 3rd International Conference on Cloud Computing*, 2010.

[12] Ekpe Okorafor, "A Fault-tolerant High Performance Cloud Strategy for Scientific Computing", *IEEE International Parallel Distributed Processing Symposium*, 2011.

[13] Sangho Yi and Derrick Kondo, Artur Andrzejak, "Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud", *IEEE 3rd International Conference on Cloud Computing*, 2010.

[14] Dilbag Singh, Jaswinder Singh, Amit Chhabra, "Evaluating Overheads of Integrated Multilevel Checkpointing Algorithms in Cloud Computing Environment", *I. J. Computer Network and Information Security*, 2012, 5, 29-38.

[15] Sangho Yi, Member, IEEE, Artur Andrzejak, and Derrick Kondo, Member, IEEE, "Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances", *IEEE Transactions on Services Computing*, vol. x, no. x, month 201X.

[16] Jose Carlos Sancho Fabrizio Petrini Greg Johnson Juan Fernandez Eitan Frachtenberg, "On the Feasibility of Incremental Checkpointing for Scientific Computing", *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)*, 2004, IEEE.

[17] Jianjian Song1, Heng Kek Choo1 and Kuok Ming Lee2, "Application-level load migration and its implementation on top of PVM", *Concurrency: Practice and Experience*, vol. 9(1), 119 (January 1997).

[18] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Csar A. F. De Rose, and Rajkumar Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", *Software: Practice 2011 - Wiley Online Library* .

[19] Rajkumar Buyya, Rajiv Ranjan and Rodrigo N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", *International Conference on High Performance Computing Simulation*, 2009. HPCS '09,P-1-11.

[20] Andreas Menychtas, Kleopatra G. Konstanteli, "Fault Detection and Recovery Mechanisms and Techniques for Service Oriented Infrastructures", *IGI Global*, 2012.

[21] Yi Hu, Bin Gong, Fengyu Wang, "Cloud Model-Based Security-aware and Fault-Tolerant Job Scheduling for Computing Grid", *The Fifth Annual ChinaGrid Conference*, IEEE, 2010.

[22] Anthony T. Velte, Toby J. Velte, Robert Elsenpeter, "Cloud Computing: A Practical Approach", *the McGraw-Hill Companies*, ISBN: 978-0-07-162695-8, MHID: 0-07-162695-6, 2010.

[23] Qingling Wang, Carlos A. Varela, Impact of Cloud Computing Virtualization Strategies on Workloads Performance", *Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, 5-8 Dec. 2011,P-130-137.

[24] N. Yadav and P.M.Khilar, "Hierarchically Adaptive Distributed Fault Diagnosis in Mobile Adhoc Networks using Clustering", *in proc. of International Conference on Industrial and Information System 2010 (ICIIS 2010)*, NITK, Surathkal, Karnatak, India during 29th July to 1stAugust 2010.

[25] Pabitra Mohan Khilar, Jitendra Kumar Singh, Sudipta Mahapatra, "Design and Evaluation of a Failure Detection Algorithm for Large Scale Ad Hoc Networks Using Cluster Based Approach", *IEEE International Conference on Information Technology*,10.1109/ICIT.2008.72.

[26] Martin Randles, Enas Odat, David Lamb, Osama Abu- Rahmeh and A. Taleb-Bendiab, "A Comparative Experiment in Distributed Load Balancing", *Second International Conference on Developments in Systems Engineering* ,10.1109/DeSE.2009.20,P-258- 265.

[27] Martin Randles, David Lamb, A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2010, P: 551- 556.

[28] A.Khiyaita , M.Zbakh, H. El Bakkali ,Dafir El Kettani , Load Balancing Cloud Computing : State of Art", *National Days of Network Security and Systems (JNS2)*, 2012, 10.1109/JNS2.2012.6249253, P: 106- 109.

[29] Guohong Cao, Student Member, IEEE, and Mukesh Singhal, Member, IEEE, "On Coordinated Checkpointing in Distributed Systems", *IEEE Transactions on Parallel and Distributed System*, Vol. 9, No. 12, December 1998.

[30] Neeraj Rathore, Dr. Inderveer Chana , "A Cognitive Analysis of Load Balancing and job migration Technique in Grid", *World Congress on Information and Communication Technologies (WICT)*, 2011,P 77-82, ISBN 978-1-4673-0127-5.

[31] A.K, M.Z, EL Bakkali, Dafir EL Kettani, "Load Balancing Cloud Computing : State of Art", ISBN 978-1-4673-1053-6 , IEEE 2012.

[32] Jinhua Hu ,Jianhua Gu, Guofei Sun, Tianhai Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing ", *3rd International Symposium on Parallel Architectures, Algorithms and Programming* , 978-0-7695-4312-3/10, 2010 IEEE,DOI 10.1109/PAAP.2010.65.

[33] Xiaona Ren , Rongheng Lin, Hua Zou, "A Dynamic Load Balancing Strategy for Cloud Computing Platform Based on Exponential Smoothing Forecast", *Proceedings of IEEE CCIS2011* , 978-1-61284-204-2/11, 2011 IEEE.

[34] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops* , 2010, pp. 551556.

[35] Derek L. Eager, Edward D. Lazowska, Jhon Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed ", *IEEE Transactions on Software Engineering* , Vol. SE-12,No 5,May 1986.

[36] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Csar A. F. De Rose, and Rajkumar Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", *Software: Practice 2011 - Wiley Online Library.*

[37] Rajkumar Buyya, Rajiv Ranjan and Rodrigo N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", *International Conference on High Performance Computing Simulation*, 2009. HPCS '09,P-1-11.

[38] Francesc J. Ferri and Enrique Vidal, "Case-Studies on Average-Case Analysis for an Elementary Course on Algorithms", *IEEE TRANSACTIONS ON EDUCATION*,VOL. 42, NO. 2, MAY 1999.

[39] Karp, R.M., Introduction to randomized algorithms,Discrete Applied Mathematics", *Computer Science Division,University of California Berkeley,International Computer Science Institute,Berkeley*, CA 94704, USA .

Dissemination of Work

1. Pranesh Das and Pabitra Mohan Khilar, "VFT: A Virtualization and Fault Tolerance Approach for Cloud Computing", *IEEE International Conference on Information and Communication Technology(ICT 2013), Norul Islam University,Kanyakumari,Tamilnadu*, April 11-12, 2013. (In press)

2. Pranesh Das and Pabitra Mohan Khilar,"LBVFT: A Load Balancing Technique for Virtualization and Fault Tolerance in Cloud Computing ",*International Journal of Computer Application(IJCA)*,Vol 69,Issue 28,Page 14-18,2013.(Published)

3. Pranesh Das and Pabitra Mohan Khilar, "A Randomized Searching Algorithm and its Performance Analysis with Binary Search and Linear Search Algorithms ", *The International Journal of Computer Science and Application(TIJCSA)*, Vol 1,No 11, 2013. (Published))